Session mobility in multimedia services enabled by the cloud and peer-to-peer paradigms

Dan Johansson

Department of Computer Science, Electrical and Space Engineering
Luleå University of Technology
Skellefteå, Sweden
Email: dan.johansson@ltu.se

Abstract—Services based on new information technology offering audio and video modality are viewed among the most important today. When users change location or device there is a need to keep these media sessions active.

The main purpose of this article is to present a lightweight framework that allows session mobility through making profit of the cloud and peer-to-peer paradigms, while at the same time fulfilling the prevailing requirements for new session mobility multimedia services.

We demonstrate the principles of the framework by creating a real prototype, allowing combined video and audio sessions to be migrated between devices, thus showing successful implementation of session mobility meeting requirements such as low degree of service provider dependency, no changes to current common network infrastructure, dealing with privacy issues, providing flexibility and low cost, and letting the user control when and where to migrate the sessions.

It is our belief that relatively new paradigms such as peer-to-peer computing and cloud services could enhance and support flexible session mobility in a mobile use context.

I. INTRODUCTION

Mobile IT usage has become an omnipresent part of our modern society. Recent statistics show that the estimated number of Internet users exceeds 2 billions, and that there are more than 5 billion mobile phone subscriptions throughout the world [1].

New paradigms have emerged that fundamentally change past infrastructures. Peer-to-peer (P2P) technology [2] connects devices and applications in a decentralized manner, allowing them to exchange information and share resources through the creation of overlay networks lacking coordination from a central server. The Cloud paradigm [3] adds internet-accessible, on-demand services in the form of software, platforms, infrastructure and hardware resources.

Services based on new information technology offering audio and video modality are viewed among the most important today [4]. Labels like Triple play, denominating services based on video, audio and data, are complemented and extended by Quad play, adding mobility [5].

Traditionally, we count four types of mobility [6]: terminal mobility, allowing a device to change location and still be able to communicate; personal mobility (or user mobility), when a user can keep his or her user identity irrespective of terminal or network; service mobility, making a particular service accessible by the user, regardless of terminal or network, and

lastly session mobility (or continuous user mobility), letting the user change location or device and still be able to keep media streams active.

In combining session mobility with multimedia services, we have laid a foundation for creating applications that can satisfy the needs and demands of today's users.

Within the MOSA [7] project, we have explored how mobile e-services can emerge from new technical platforms and paradigms that offer increased dynamics, accessibility and coverage. The main purpose of this article is to present a lightweight framework called MVCS that allows session mobility through making profit of the cloud and peer-to-peer paradigms, while at the same time fulfilling the prevailing requirements for session mobility multimedia services. A real system implementation is created to evaluate the solution.

The paper is organized as follows: in Section II we present an overview of related work to the requirements for multimedia services providing session mobility given in Section III. The description of our system implementation is covered in Section IV. Section V contains the evaluation of our system followed by our results and, in Section VI, our conclusions.

II. RELATED WORK

There are many systems and architectural proposals for session mobility. They can be categorized into different taxonomies, depending on, for instance, centricity, scope and protocols used.

An architecture for session mobility can be either network centric or device centric [5]. Using the former approach, the migration is at large handled by the network. The user and/or the system only asks the network to initiate and take care of the session migration, from one device to another. The role of the user and the device is restricted to just providing some preferences and maybe advertise themselves. The network might even be given the responsibility of choosing the best target destination for the session. As the network centric approach is associated with vast complexity within the network, and also high deployment cost, it might be most suitable for small, localized session mobility-based systems. A device centric approach is characterized by putting the mechanisms and functions for realizing session mobility on the device, using the network as a more or less unintelligent infrastructure for data transfer. The devices, with or without interference

from the user, are incumbent of gathering information from the network to initiate, suspend, migrate, resume and end sessions. Obviously this requires more complex and capable applications and devices, but network dependency is vastly reduced compared to network centric solutions.

Looking at different protocols, solutions based on the Session Initiation Protocol (SIP) [8] have become popular. Being adopted as the signaling protocol of VoIP and 3GPP [9], SIP has a strong position within the session mobility area. The protocol provides tools for controlling multimedia sessions, e.g. audio and video, over the Internet Protocol (IP). Architectures are based upon User Agent Clients and Servers, exchanging SIP messages to initiate, modify and terminate sessions. Proxies, Registrars and Redirects respectively negotiate initiation of sessions on the users behalf, registers sessions and redirects them, should a migration be needed. Displayed in [10] and [11] are two examples of SIP based solutions. One drawback of session mobility through SIP is that the signaling is relatively heavy and requires text messages of considerable size. Some kind of proxy and/or redirect must also be included for a SIP based system to work.

In [12] a SIP solution is extended with a web-based communication system. Using a web server and a media server to make available and support media, piecewise protocols are combined within a so called network box resulting in fewer SIP messages and less processing overhead. The solution is however early work and not yet implemented as a real prototype.

Zugenmaier et al [13] present how to achieve anchorless session mobility, that is creating session mobility that does not rely on home agents, as e.g. in Mobile IP, which cannot support session mobility without extensions, or the SIP Proxy. The solution is inspired by the Host Identity Protocol (HIP). HIP separates the end-point identifier and locator roles of IP addresses, thus providing multihoming, identifying the user rather than the Internet Service Provider. A theoretical description of a protocol incorporating these ideas is presented. No simulations or measurements are presented, the emphasis is rather on how to make the session migration as secure as possible, though the authors admit that more formal proof is needed to confirm that the security goals are fulfilled. As the protocol adds an additional layer between transport and network layers, this forces all communication partners to support the protocol.

Finally, there are also middleware solutions, where certain session or relocation managers are installed on all devices. Barisch et al [14] propose a framework that can cover different applications, does not involve a communication partner and allows session mobility. The TCP based Middleware solutions are obviously very device centric in their nature.

III. REQUIREMENTS

Spedalieri et al [4] list requirements for multimedia content management in new service platforms. Requirements are based on interviews with experts within the industry. Results show that session mobility is seen an interesting service, but often

hard to achieve with existing architectures. As pointed out in [14], many existing solutions depend on a service provider and the exchange of signaling messages, thus requiring that the service provider supports such session transfers. No modifications should have to be done on the communication partner's network (i.e. no new hardware or demands for SIP support or reconfiguration of routers). Privacy also becomes an issue, as the service provider is explicitly informed about a session migration taking place, often with details of how and whereto. Flexibility, convenience and low cost are of high importance when creating systems encompassing this kind of mobility [4]. User control over the session migration is also stressed. The process should not be automatic but initiated by the user.

The five explicit requirements for our prototype could therefore be summarized in the following points, showing issues that have to be addressed:

- 1) Low degree of service provider dependency
- 2) No changes to current common network infrastructure
- 3) Privacy
- 4) Flexibility, convenience and low cost
- 5) User control over the session migration

IV. SYSTEM DESCRIPTION

Our solution for session mobility, the MOSA Video Conference Service (MVCS), consists of two major components, namely the MVCS Application and the MVCS Server. The MVCS Application allows the user to set up and take part in video conferences with other MVCS users, while the MVCS Server works as a back-end and repository. In addition, the system makes use of three third-party cloud services, more exactly peer assisted networking, an authentication service and a cloud-to-device messaging framework. Communication is carried out through facultative network infrastructures, e.g. WiFi, Edge or 3G, and thus not limited to local networks. Figure 1 contains an overview of our architecture.

A. Authentication service

The authentication is managed by a third-party cloud service, in this particular case the Google ClientLogin [15]. First, the user supplies the MVCS Application with a login and a password, corresponding to the user credentials of a beforehand-created Google account. The MVCS Application requests a token with a ClientLogin from the Google accounts authorization and a response is subsequently sent to the application. If the challenge was met by a success (HTTP 200) the response will contain an authorization token that will be valid through out the session.

B. MVCS Application

The MVCS Application allows the user to set up and take part in video conferences with other MVCS users. The application exists in two variants, a full version (for stationary or semi-stationary computers) and a mobile version, the latter intended to run on smartphones.

Fig. 1. MVCS Architecture

1) *Implementation*: The application was implemented using the Adobe Flash Builder framework [16], relying on ActionScript and MXML for creating interfaces and interactivity. The Adobe AIR runtime enables building web applications that execute as standalone applications, thus making it possible to install the applications as native apps on different devices.

2) *GUI*: Figure 2 shows a screenshot of the lobby. The lobby provides a graphical representation of available conference rooms and shows information about the number of users present in each room. There are buttons for creating new conference rooms, joining existing rooms and refreshing the list, as well as the "Go mobile" button, which allows the user to continue working with the MSVC on a handheld device.

After joining a room, the view switches to show the user in a small broadcasting window, and other participating users in a bigger, receiver window. Through this GUI, shown in Figure 3, the user can communicate with the others through text (chat), video and audio.

By clicking the "Go mobile" button, the session will be

Fig. 2. MVCS Lobby

Fig. 3. MVCS Conference room

transferred to a handheld device, registered by the user with the cloud-to-device messaging server. The program will start automatically and the session will be set up. Also, the chat history is saved. A screenshot is shown in Figure 4.

C. MVCS Server

The MVCS Server is an important component of the system. It works as a back-end, as well as a repository, offering centralized storage of data to be shared between the applications and the third-party cloud services. An Xampp web server allows the rest of the system to communicate with the central database through a back-end consisting of Hypertext Preprocessor (PHP) files. Most calls consist of information retrieval or data storage, but the back-end also handles requests to the cloud services.

The repository contains three tables. The first one keeps track of the devices, both mobile and stationary, as well as semi-stationary or nomadic, e.g. laptops. Their IP addresses are stored along with information about who owns them (in the form of unique Google usernames) and arbitrary device names. Another table contains information about the chat

Fig. 4. MVCS Mobile Application

rooms, with groupnames, peer group identifications and chat history. The last table lists the users, storing their unique Google usernames as primary key and appendant cloud-to-device messaging registration identifications and information about a possible ongoing session.

D. Cloud to device messaging framework

To be able to send data from the MVCS Server to applications installed on Android devices, we make use of the Android Cloud to Device Messaging (C2DM) framework [17]. Joining the system for the first time, the mobile device registers with the messaging service through the MVCS Application by sending an intent including the sender identification (i.e. a developer account) and the application identification (the package name taken from the application manifest). If successful, C2DM will return a registration identification to the application, which in turn sends it to the MVCS Server, storing it in the repository. This routine is only carried out during the set up of the system. After registration, the identification is always retrieved from the MVCS Server when needed.

When moving the session from the computer to the mobile device, the user first interacts with the MVCS Server, sending a request to go mobile. The MVCS Server contacts the C2DM server, providing the registration identification that links the server to the application and its mobile host. Should the mobile device be inactive, the C2DM server will store the message, otherwise it will be sent to the device instantly. When receiving the message, the application wakes up and starts to process the message, contacting the MVCS Server directly. In this particular case, the MVCS Application starts and automatically opens the lobby (or a login window, should the user not

TABLE I
DEVICES USED IN TESTBED

Name	Type	Specification
L1	Laptop	HP, Intel Core Duo 2.26 GHz, 2 GB RAM, Windows 7
S1	Smartphone	Google Nexus One, Android 2.2.3
L2	Laptop	HP, Intel Core Duo 2.26 GHz, 2 GB RAM, Windows Vista
S2	Smartphone	HTC Desire, Android 2.2

have an active authentication token registered with the device), where the session can be resumed.

E. Peer assisted networking

Setting up the actual video streams, the MVCS uses Real Time Media Flow Protocol (RTMFP). The protocol is developed by Adobe and supported by all computers using Flash Player 10 or newer. It allows direct communication in peer-to-peer style between the different copies of the MVCS applications. The protocol is UDP based, which makes it suitable for quick and efficient video and audio transmission at the expense. Minimal latency is achieved as lost data is not re-transmitted, as would be the case with TCP. As audio data transmission is prioritized over e.g. video and text messages, bandwidth limitations does not affect the quality of the session in the same way as it would in a system without data prioritization.

When starting a MVCS Application, it connects to the Adobe Cirrus server. The server acts as a rendezvous making it possible to join peer-to-peer groups (corresponding to conference rooms in the MVCS). When the applications are linked together in the peer network, they function as endpoints that can send and receive audio and video streams directly between each other. The streams does not have to be routed via some central server, and it does not matter if the communicating devices are located behind NATs or firewalls as long as the firewall allows outgoing UDP traffic. Communication is protected through 128-bit AES. Unique identifications for keeping track of groups and sessions are emitted by the Cirrus server.

F. Testbed implementation

The system was implemented using two laptops and two smartphones, all connected to an 802.11 WiFi network. The laptops were running different versions of Windows: Vista and Windows 7 respectively. The smartphones both used the Android operative system. In an additional test, the HCT smartphone was connected to a EDGE network, while the Google smartphone was still connected to the original 802.11 WiFi. Table I summarizes the testbed setup.

V. EVALUATION AND RESULTS

A. Scenario

The use scenario contains all the devices listed in table I. Devices L_1 and S_1 , running MVCS Applications A_1 , and A'_1 (the latter one representing the mobile version) belong to user

Fig. 5. The detailed sequence of steps carried out during the session migration scenario

U_1 while devices L_2 and S_2 , running MVCS Applications A_2 and A'_2 , belong to user U_2 .

In the scenario, U_2 starts the MVCS and creates a conference room. U_1 logs on to the system, joins the chat room via L_1 and eventually decides to move the session to S_1 , continuing the video conference from the smartphone. User U_2 then migrates her session to S_2 , after which U_1 moves back to L_1 , i.e. the laptop from where she started the session. This scenario could thus simulate two co-workers communicating while switching between being stationary and on the move, maintaining their ongoing video and audio session. Figure 5 shows the sequence of steps carried out.

B. Measurements

Figure 6 shows the time it took to migrate the session from A_1 to A'_1 . The migration sequence is split in three steps: the first one shows the time it took the MVCS to send the push message to A'_1 and start the application, the second step is to authenticate and show the lobby after providing user credentials, while the third and final step shows the time it takes entering the conference room and restoring the actual

conference session. The results show that a complete migration of a session from A_1 to A'_1 takes approximately 14 seconds. The time for moving back to the original device A_1 and continuing the session is negligible, less than 1 second and approximately 1 second respectively (mean values). This has to do with the fact that the application is still running in the background at the original device (L_1) with audio and video streams paused.

C. Meeting of requirements

The MVCS has a low degree of service provider dependency. The major functions are located within the MVCS Server and Applications, which are owned by the user/users (be they users or groups within an organization). Google services are harnessed for authentication and contacting the mobile device, while the Adobe Cirrus server is used for adding peer-to-peer functionality, minimizing centralized communication such as the need for a media server. Audio and video streaming is carried out directly between the peers.

Using the MVCS requires no changes to the underlying IT-infrastructure. A mobile device that supports Android Air

Fig. 6. Migration time (mean values) moving session from A to A'

and push notification (which is the case with devices running Android 2.2 or newer) and a PC with the Adobe Air runtime is all that is needed.

Sessions are carried out directly between peers. Conference rooms, information about users and their devices and chat history is stored on the MVCS Server, which is owned and maintained by the user/users.

The system is flexible in that way that it allows a wide range of devices (Android 2.2+ being the requirement for mobile devices, Android Air runtime for stationary and semi-stationary devices) and can be run over all kinds of IP based networks. The MVCS also deals with the double jump problem, that is when both ends of a session moves, i.e. A_1 to A'_1 and A_2 to A'_2 . Except traffic cost, the system is free to use as the cloud services are open.

The user has full control over the migration process. She actively pushes the session from her laptop to the mobile device by pressing a button. When migrating the session from the mobile device, the user can choose a new host from a list of devices running active applications. Only her own, registered devices will appear in the list.

The MVCS thus fulfill all the requirements stated in Section III.

VI. CONCLUSION

The main purpose of this article was to present a lightweight framework that allows session mobility through making profit of the cloud and peer-to-peer paradigms, while at the same time fulfilling the prevailing requirements for session mobility multimedia services.

A prototype was created, allowing video, audio and chat sessions to be migrated from stationary and semi-stationary devices to smartphones, thus demonstrating successful implementation of session mobility meeting requirements such as low degree of service provider dependency, no changes to

current common network infrastructure, dealing with privacy issues, providing flexibility and low cost and letting the user control when and where to migrate the sessions.

It is our belief that relatively new paradigms such as peer-to-peer computing and cloud services could enhance and support flexible session mobility in a mobile use context.

Future work would focus on refining the architecture, for instance to support parallel applications and/or split sessions between devices. Another task would be to explore the mobility type, extending mobility of sessions to encompass the whole application, thus enabling application mobility, which is still a comparably unexplored mobility area.

Also, the new, still emerging, standard for HTML5 contains many concepts that could form the next step in the convergence of multimedia, cloud and peer-to-peer technology, e.g. elements that support audio and video distribution, APIs for peer-to-peer connectivity, enhanced in-browser capabilities, local databases etc. It should be explored how session mobility could be supported and enriched through this interesting and rapidly evolving web infrastructure.

ACKNOWLEDGMENT

This article has been compiled as part of the Mobile and Open Service Access project [7], which is supported by EU structural funds (Objective 2).

The author would like to thank Jonas Öhlund and Robert Wiklund for valuable programming efforts.

REFERENCES

- [1] ITU, "Free statistics", <http://www.itu.int/ITU-D/ict/statistics/>, May 2011
- [2] J. F. Koegel Buford, H. H. Yu, and E. K. Lua, "P2P networking and applications", Amsterdam: Elsevier/Morgan Kaufmann; 2009.
- [3] B. P. Rimal and I. Lumb, "A Taxonomy and Survey of Cloud Computing Systems", Fifth International Joint Conference on INC, IMS and IDC, 2009, NCM '09, pp.44-51, 25-27 Aug. 2009
- [4] A. Spedalieri, A. Sinfreu, G. Sisto, P. Cesar, I. Vaishnavi, and D. Melpignano, "Multimedia content management support in next generation service platforms", In Proceedings of the 3rd international conference on Mobile multimedia communications (MobiMedia '07), ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Article 14, 7 pages, 2007.
- [5] S. Mate, U. Chandra, and I. D. D. Curcio, "Movable-multimedia: session mobility in ubiquitous computing ecosystem", In Proceedings of the 5th international conference on Mobile and ubiquitous multimedia (MUM '06), ACM, New York, NY, USA, Article 8, 2006.
- [6] ITU, "ITU-T Recommendation H.510, Mobility and Collaboration procedures", Mobility for H-Series multimedia systems and services, http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T, 2002.
- [7] MOSA, "Mobile and Open Service Access", <http://www.mosaproject.org/>, 2008.
- [8] H. Schulzrinne and E. Wedlund, "Application-layer mobility using SIP", SIGMOBILE Mob. Comput. Commun. Rev. 4, 3, 47-57, July 2000.
- [9] 3GPP, <http://www.3gpp.org/>, 2011.
- [10] R. Shacham, S. Schulzrinne, S. Thakolsri, and W. Kellerer, "The virtual device: expanding wireless communication services through service discovery and session mobility", Wireless And Mobile Computing, Networking And Communications, 2005. (WiMob'2005), IEEE International Conference on , vol.4, no., pp. 73- 81 Vol. 4, 22-24 Aug. 2005.
- [11] M-X. Chen and F-J. Wang, "Session mobility of SIP over multiple devices", In Proceedings of the 4th International Conference on Testbeds and research infrastructures for the development of networks & communities (TridentCom '08), ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Article 23, 9 pages, March 2008.

- [12] S. Shanmugalingam, N. Crespi, P. Labrogere, "User mobility in a Web-based communication system", IEEE 4th International Conference on Internet Multimedia Services Architecture and Application (IMSAA), pp.1-6, 15-17 Dec. 2010.
- [13] A. Zugenmaier, J. Laganier, A. Prasad, and K. Slavov, "Anchor-less secure session mobility", In Proceedings of the 15th international conference on Security protocols, Bruce Christianson, Bruno Crispo, James A. Malcolm, and Michael Roe (Eds.), Springer-Verlag, Berlin, Heidelberg, 104-109, 2007.
- [14] M. Barisch, J. Kögel, and Sebastian Meier, "A Flexible Framework for Complete Session Mobility and Its Implementation", In Proceedings of the 15th Open European Summer School and IFIP TC6.6 Workshop on The Internet of the Future (EUNICE '09), Miquel Oliver and Sebastia Sallent (Eds.). Springer-Verlag, Berlin, Heidelberg, 188-198, 2009.
- [15] Google Code, "ClientLogin for Installed Applications - Authentication and Authorization for Google APIs - Google Code", <http://code.google.com/intl/sv-SE/apis/accounts/docs/AuthForInstalledApps.html>, 2011.
- [16] Adobe, www.adobe.com, 2011.
- [17] Google Code, "Android Cloud to Device Messaging Framework Google Projects for Android", <http://code.google.com/intl/sv-SE/android/c2dm/index.html>, 2011.